

Framework for the Interoperability of Software Engineering Metamodels

Muhammad Atif Qureshi



A thesis submitted for the degree of
Doctor of Philosophy
University of Technology, Sydney

July 2014

Declaration

This is to certify that this thesis comprises my original work toward the Doctor of Philosophy degree and due acknowledgement has been made in the text of all other materials used. The work has not been submitted previously, in whole or part, to qualify for any other academic award. Any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Muhammad Atif Qureshi

(July 2014)

*Dedicated to my mother. At her knees my education commenced and to her I
owe all that I am or hope to be.*

Abstract

A model, represented as a concrete artefact, is an abstraction of reality according to a certain conceptualization. A model can support communication and analysis about relevant aspects of the underlying domain. A model must be expressed in some language and such languages are defined using metamodels. Many metamodels have been proposed and used in the software engineering literature. Some define modelling languages that are general in nature but the literature of modelling is dominated by domain-specific modelling languages or metamodels. Most of these metamodels have been developed independently from each other and any shared concepts are only accidental. Widespread adoption of these metamodels is hindered by differences between metamodels' concepts. Using more than one modelling language during software development requires some sort of interoperability between the metamodels of those modelling languages. This interoperability is also required to allow mappings between models developed using different modelling languages.

These metamodels are not static in nature and are continuously evolving. This evolution has increased their size and complexity over time. This complexity increases when more than one metamodel is used

during software development. Interoperability of a pair of metamodels can reduce their joint size and complexity (elaborated in detail in Chapter 7). The need for interoperability between metamodels is also raised by many research communities.

In this thesis, we have developed a framework that can be used for metamodel interoperability. The framework compares metamodel elements based on their syntax, semantics and structure. The semantics of metamodel elements are further investigated for *linguistic* and *ontological* semantics.

Since terms such as *interoperability*, *bridging*, *merging* and *mapping* have all been used, often loosely, with reference to metamodel compatibility, we will define these terms under the generic term *harmonization*.

Metamodels share some similarities with other domains, e.g. ontologies and schemas. In this thesis, we have also explored the techniques available in these domains that might be useful for metamodel interoperability. We have applied our framework to different metamodels and have shown how metamodels can be used in an interoperable fashion.

The results achieved are analysed and we have shown how interoperability of metamodels can reduce their size and their joint complexity, hence making them easier to understand and use.

Acknowledgement

Firstly, my greatest regards to the Almighty Allah for bestowing upon me the blessings, throughout my life, that make me able to achieve this milestone.

There are no words I could possibly write to articulate my gratitude to my mother, whose never ending prayers followed me everywhere and gave me the courage to face the complexities of life and complete this project successfully.

I must express my gratitude to my wife for her continued support and encouragement. Completing this work would have been all the more difficult without her support. She was undoubtedly the best partner I could have had during the ups and downs of my journey. Also, special thanks to my daughters, who always give me a new hope and strength.

I am immeasurably proud of my brothers and my sisters. I am grateful to them, for all their love, care and everlasting support in these many years of physical distance.

I would like to thank my supervisor, Prof. Brian Henderson-Sellers, for the patient guidance, encouragement and advice he has provided throughout my time as his student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. I would also like to thank Dr Bruce Howarth for his help regarding proof reading and improving the presentation of this thesis.

This thesis is dedicated to all these people. I could not have completed my research without the support of all these wonderful people!

Publications

Publications arising from this thesis.

1. Qureshi, M. A. (2011), Interoperability of software engineering metamod-els. In Proceedings of the 2011th international conference on Models in Software Engineering (MODELS'11), Jorg Kienzle (Ed.). Springer-Verlag, Berlin, Heidelberg, 12-19.
2. Henderson-Sellers, B., Qureshi, M. A., and Gonzalez-Perez, C. (2012). Towards an interoperable metamodel suite: Size assessment as one input. International Journal of Software and Informatics, 6(2):111124.
3. Qureshi, M. A. (2012), Interoperability of software Engineering Metamod-els: Lessons Learned., in Ulrich W. Eisenecker and Christian Bucholdt, ed., Software Language Engineering (Doctoral Symposium), CEUR-WS.org, pp. 3-10.

Contents

1	Introduction	1
1.1	Background	2
1.2	Motivation	4
1.3	Objectives	6
1.4	Scope	7
1.5	The Approach	8
2	Model-Driven Engineering	12
2.1	Models	14
2.1.1	Descriptive and Specification Models	14
2.2	Modelling Languages	18
2.3	Metamodels	21
2.4	Model-Driven Architecture (MDA)	26
2.4.1	Linguistic and Ontological Metamodelling	30
2.5	Conclusion	33
3	Metamodels and Ontologies	35
3.1	Ontologies in Software Engineering	37
3.1.1	Ontologies and Metamodels	41
3.2	Conclusion	47

4	Similarity Matching Techniques	48
4.1	Harmonization of Terms	49
4.1.1	Matching	50
4.1.2	Interoperability	53
4.1.3	Merging and Integration	54
4.1.4	Alignment	56
4.1.5	Bridging	56
4.1.6	Mapping	57
4.2	Similarity Techniques in Models and Metamodels	59
4.3	Similarity Techniques in Ontologies	62
4.4	Similarity Techniques in Schemas	64
4.5	Similarity Techniques in Business Process Models	66
4.6	Conclusion	68
5	The Interoperability Framework	70
5.1	Selection of Metamodels	73
5.2	Linguistic Analysis	74
5.2.1	Syntactic Analysis	74
5.2.2	Semantic Analysis	75
5.2.3	Structural Analysis	77
5.3	Ontological Analysis	81
5.3.1	Linguistically Same Concepts	81
5.3.2	Linguistically Similar Concepts	82
5.3.3	Linguistically Different Concepts	82
5.3.4	Holonyms and Meronyms	83
5.3.5	Hypernyms and Hyponyms	84
5.4	Interoperability	86
5.5	Conclusion	86

6	Application of the Framework	88
6.1	Example 1: BPMN and OSM	89
6.1.1	BPMN	89
6.1.2	OSM	90
6.1.3	Syntactic Similarity of BPMN and OSM	91
6.1.4	Semantic Similarity of BPMN and OSM	93
6.1.5	Structural Similarity between BPMN and OSM	93
6.1.6	Ontological Analysis	95
6.1.7	Interoperability of BPMN and OSM	102
6.2	Example 2: MAS Metamodels	105
6.2.1	ADELFE	106
6.2.2	Gaia	107
6.2.3	PASSI	108
6.2.4	Syntactic Similarity of ADELFE, Gaia and PASSI	109
6.2.5	Semantic Similarity of Gaia, ADELFE and PASSI	111
6.2.6	Ontological Analysis	113
6.2.7	Interoperability of Gaia, ADELFE and PASSI	118
6.3	Example 3: MaSE and Prometheus	120
6.3.1	MaSE	120
6.3.2	Prometheus	120
6.3.3	Syntactic Similarity of MaSE and Prometheus	121
6.3.4	Semantic Similarity of MaSE and Prometheus	123
6.3.5	Ontological Analysis	124
6.4	Example 4: Ecore and UML	128
6.4.1	Ecore and UML	128
6.4.2	Syntactic Similarity of ECore and UML	129
6.4.3	Semantic Similarity of ECore and UML	131

6.4.4	Structural Similarity of ECore and UML	132
6.4.5	Ontological Analysis	132
6.5	Example 5: SPEM and BPMN	137
6.5.1	SPEM and BPMN	137
6.5.2	The Scenario	141
6.6	Conclusion	144
7	Quality Assessment	145
7.1	Introduction	145
7.2	Quality Assessment	145
7.3	Size and Complexity Measures	149
7.4	Conclusion	152
8	Conclusions	154
8.1	Lessons Learned	155
8.1.1	Syntactic Matching	155
8.1.2	Semantics Matching	156
8.1.3	Comparing Structures	158
8.1.4	Automation	160
8.2	Future Research Plans	162

List of Figures

2.1	Homomorphism between a model and the SUS (González-Pérez and Henderson-Sellers, 2009)	15
2.2	Type and token models (Kühne, 2005)	17
2.3	Relationship between a model and the SUS	18
2.4	Relationship between a language, model and model-unit-kind (Gonzalez-Perez and Henderson-Sellers, 2007)	19
2.5	The concept of notation (Gonzalez-Perez and Henderson-Sellers, 2007)	20
2.6	Misleading interpretation of a metamodel	22
2.7	Relationships among model, modelling language and metamodel (Kühne, 2006a).	22
2.8	Bézivin and Gerbe (2001)’s interpretation of metamodel	23
2.9	Relationship between model, metamodel and modelling language, adapted from (Gonzalez-Perez and Henderson-Sellers, 2007) and (Kühne, 2006b)	25
2.10	OMG four-layer modelling infrastructure, (Seidewitz, 2003)	26
2.11	<i>instanceOf</i> relationship between different levels of models using strict metamodeling	27
2.12	Value of an attribute at instantiation	28

2.13	Multiple <i>instanceOf</i> relationships between class and object . . .	28
2.14	Linguistic and ontological relationships, adapted from (González-Pérez and Henderson-Sellers, 2006)	31
2.15	Re-arrangements of linguistic and ontological relationships, adapted from (González-Pérez and Henderson-Sellers, 2006)	31
2.16	Example of powertype pattern (Henderson-Sellers and González-Pérez, 2005)	33
3.1	Roles of ontologies in software engineering (Happel and Seedorf, 2006)	39
3.2	Three-level ontology architecture suggested by the Ontology Definition Metamodel of the Object Management Group (after Figure 12 in (Henderson-Sellers, 2011))	41
3.3	Three domains defined by the category of users. For models created in any one of these domains, a modelling language (ML) is used. That ML is often the same for the three layers and can be defined in one of a number of ways (after Figure 6.1 in (Henderson-Sellers, 2012))	42
3.4	Relationship between metamodels and domain ontologies	44
4.1	Harmonization and matching	50
4.2	The MDA architecture from CIM (computationally independent model) to PIM (platform independent model) to PSM (platform specific model)	52
4.3	Metamodel transformation	53
4.4	Metamodel interoperability	54
4.5	Merging of concepts from BPMN and OSM	55
4.6	Bridging of concepts in BPMN and OSM	57

4.7	Mapping of concepts in BPMN and OSM	58
4.8	Ontology merging method (Stumme and Maedche, 2001)	62
4.9	Classification of schema matching techniques (Rahm and Bernstein, 2001)	65
5.1	Metamodel interoperability framework	72
5.2	Structural neighbours of a class C (de Sousa Jr et al., 2009) . . .	78
5.3	Activity and ActivityResource as holonym/meronym in BPMN (OMG, 2011a)	83
5.4	<i>Event</i> as hypernym in BPMN (OMG, 2011a)	84
6.1	Participant in OSM	96
6.2	Participant in BPMN	96
6.3	Property in BPMN	97
6.4	Property in OSM	97
6.5	Performer in BPMN	98
6.6	Relationship in BPMN	100
6.7	Interaction specification in BPMN	101
6.8	Interoperability of BPMN and OSM (dashed lines show mappings between elements of the two metamodels)	104
6.9	MAS metamodel adopted in ADELFE (Bernon et al., 2003) . .	106
6.10	MAS metamodel adopted in Gaia (Bernon et al., 2003)	108
6.11	MAS metamodel adopted in PASSI (Bernon et al., 2003)	109
6.12	Mapping/merging of concepts to our proposed merged metamodel	119
6.13	MAS metamodel adopted in OMaSE	121
6.14	MAS metamodel adopted in Prometheus	122
6.15	Merged metamodel for MaSE and Prometheus	126
6.16	Mapping from MaSE to merged metamodel	127

6.17 Mapping from Prometheus to merged metamodel	127
6.18 Ecore metamodel	129
6.19 UML class diagram metamodel	130
6.20 Merged metamodel for Ecore and UML2CD	136
6.21 SPEM metamodel (partial)	139
6.22 BPMN metamodel (partial)	140
6.23 An example scenario using the merged metamodel	143
7.1 Improvements in precision	148
8.1 Number of concepts having structural similarity of some degree before using Equation 5.3 and after	161

List of Tables

3.1	Similarities and differences between metamodels and ontologies. .	45
3.2	Definitions of the terms used in table 3.1.	46
5.1	Edit distance values between <i>Process</i> and <i>Activity</i>	76
5.2	SSM values between <i>Process</i> and <i>Activity</i>	77
6.1	Syntactic similarity between BPMN and OSM	92
6.2	Semantic similarity between BPMN and OSM	94
6.3	Structural similarity (structSim) between BPMN and OSM . . .	94
6.4	Candidate concepts of BPMN and OSM for interoperability . . .	95
6.5	Matched concepts of BPMN and OSM after ontological analysis .	102
6.6	Syntactic similarity between Gaia and ADELFE	109
6.7	Syntactic similarity between Gaia and PASSI	110
6.8	Syntactic similarity between ADELFE and PASSI	110
6.9	Syntactically same concepts among Gaia, ADELFE and PASSI .	110
6.10	Semantic similarity between Gaia and ADELFE	111
6.11	Semantic similarity between Gaia and PASSI	112
6.12	Semantic similarity between ADELFE and PASSI	112
6.13	Candidate concepts for merging among Gaia, ADELFE and PASSI	112
6.14	SSM between MaSE and Prometheus	122

6.15	Synonyms for concepts in MaSE	123
6.16	Candidate concepts of MaSE and Prometheus for merging	123
6.17	Syntactic similarity between Ecore and UML2CD	131
6.18	Semantic similarity between Ecore and UML2CD	131
6.19	Structural similarity between Ecore and UML2CD	133
6.20	Candidate concepts between Ecore and UML2CD for interoper- ability	134
6.21	Similarities between BPMN and SPEM	141
7.1	Precision and recall of matching using old techniques	147
7.2	Precision and recall of matching using the interoperability frame- work	148
7.3	Size and complexity improvement	152
8.1	Effect of Equation 8.1 on structural similarity	161
8.2	BPMN Concepts List	186
8.2	BPMN Concepts List	187
8.2	BPMN Concepts List	188
8.2	BPMN Concepts List	189
8.2	BPMN Concepts List	190
8.3	ECore Concepts List	190
8.4	UML Class Diagram Concepts List	190
8.4	UML Class Diagram Concepts List	191
8.5	OSM Concepts List	191
8.6	Concepts List of Gaia, ADELFE and PASSI	192
8.7	OMaSE Concepts List	193
8.8	Prometheus Concepts List	193
8.9	BPMN Synonyms List	194

8.9 BPMN Synonyms List	195
8.10 OSM Synonyms List	195
8.11 Semantic Similarity between BPMN and OSM	196
8.11 Semantic Similarity between BPMN and OSM	197
8.11 Semantic Similarity between BPMN and OSM	198
8.11 Semantic Similarity between BPMN and OSM	199
8.11 Semantic Similarity between BPMN and OSM	200
8.11 Semantic Similarity between BPMN and OSM	201
8.11 Semantic Similarity between BPMN and OSM	202